APPLICATION NOTE

AP-285

October 1986

8207 User's Manual

Order Number: 230822-003

CHAPTER 1
INTRODUCTION

This guide is a supplement to the 8207 Data Sheet and is intended as a design aid and not a stand-alone description of the 8207. The reader should already have read and have a copy of the 8207 Data Sheet, 8206 Error Detection and Correction Unit (EDCU) Data Sheet, a microprocessor Data Sheet, or a Multibus bus specification for interfacing to the 8207, and a dynamic RAM Data Sheet(1).

The Intel 8207 Advanced Dynamic RAM Controller is a high performance, highly integrated device designed to interface 16K, 64K, and 256K dynamic RAMs to Intel microprocessors. The 8207, with the 8206, provides complete control for memory initialization, error correction, and automatic error scrubbing.

The 8207 has three speed selected versions. The -16 version is for clock speeds up to 16 MHz in "fast cycle" configurations. The -8 and -10 parts can only be used in "slow cycle" configurations and are for clock speeds up to 8 MHz and 10 MHz respectively.

NOTE:
1. All RAM cycle timings and references are based upon DRAMs from one particular vendor and will vary depending upon manufacturer.

CHAPTER 2

PROGRAMMING THE 8207

The many configurations of bus structures, RAM speeds, and system requirements that the 8207 supports require the 8207 to be programmable. The 8207 will modify its outputs to provide the best performance possible. The 8207 must be told what type of interface the memory commands will arrive on, what type of RAM (speed, refresh rate) is being used, the clock rate, and others.

The 8207 uses two means to be informed of the user's requirements. It reads in a 16-bit serial program word and examines the logic states on several input pins. The pins that are sampled for a logic level give the user options on the types of refresh and memory command input timing.

Input Pin Options

The three input pins that configure part of the 8207 are: PCTLA, PCTLB, and REFRQ. Let's examine the options in refresh types the REFRQ pin provides.

REFRESH TYPES

The 8207 gives the user a choice of the following refresh types.

1) Internal Refresh: All refresh cycles are generated internally—based on an internal programmable time.
2) External Refresh with Failsafe: If the external logic does not generate a refresh cycle within the programmed period, the 8207 will.
3) External Refresh—No Failsafe or No Refresh; All refresh cycles are generated at times by the user. This is for systems that cannot tolerate the random delay imposed by refresh (i.e. graphics memory).
4) Burst Refresh: The 8207 generates up to 128 consecutive refresh cycles and must be requested by external logic. Memory requests will be performed when the burst is completed.

The 8207 examines the state of the REFRQ pin when RESET goes inactive. This timing is shown in the "Clock and Programming Timings" waveforms in the Data Sheet.

If REFRQ is sampled active by the falling edge of RESET, the 8207's internal timer is enabled. The timer's period is determined by the CI0, CI1, and PLS bits in the program word. External refresh cycles are generated by a low to high transition on the REFRQ input. This transition, besides generating a refresh cycle, also resets the internal timer to zero. Simply tie REFRQ to Vcc if internal refresh is required.

If REFRQ is seen low at the falling edge of RESET, the internal timer is deactivated. All refresh cycles must either be done by external logic or by accessing all RAM (internal) rows within a 2 ms period.

Once the no failsafe option is programmed, the 8207 will generate a burst of up to 128 refresh cycles when the REFRQ input goes from low to high and sampled high for two consecutive clock edges. These cycles are internally counted and the 8207 stops when the refresh address counter reaches the value $XX1111111_2$ (X = don't care; see *Refresh Counter* section). If prior to the burst request the counter is at $XX1111110_2$ then only 2 refresh cycles would be generated.

For a single refresh cycle to be generated via external logic, the REFRQ input will have to go from low to high and then sample high by a falling 8207 clock edge. Since external refresh requests typically arrive asynchronously with respect to the 8207's clock, this requires the REFRQ to be synchronized to the 8207 clock when programmed in the failsafe mode. This is to ensure that the request is seen for one clock—no more, no less. If no external synchronization is performed, then the 8207 could do random burst cycles.

PROCESSOR INTERFACE OPTIONS

The PCTLA, PCTLB input pins will program the 8207 to accept either the standard demultiplexed $\overline{RD}$ and $\overline{WR}$ inputs, or to directly decode the status outputs of Intel's iAPX86, 88 family of microprocessors. The state definitions of the status lines and their timings, relative to the processor clock, differ for the 8086 family and the iAPX286 processor. Table 1 illustrates how the 8207 interprets these inputs after the PCTL pins are programmed.

If PCTL is seen high, as RESET goes inactive, an 8086 status interface is enabled. The commands arriving at the 8207 are sampled by a rising clock edge. When PCTL is low, the 80286 status and Multibus command interface is selected. These commands are sampled by the 8207 by a falling clock edge.

Table 1. Status Coding of 8086, 80186 and 80286

Status Code			Function	
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	8086/80186	80286
0	0	0	Interrupt	Interrupt
0	0	1	I/O Read	I/O Read
0	1	0	I/O Wriite	I/O Write
0	1	1	Halt	Idle
1	0	0	Instruction Fetch	Halt
1	0	1	Memory Read	Memory Read
1	1	0	Memory Write	Memory Write
1	1	1	Idle	Idle

8207 Response

8207 Command			Function	
PCTL	$\overline{RD}$	$\overline{WR}$	8086 Status Interface	Command Interface
0	0	0	Ignore	Ignore
0	0	1	Ignore	Read
0	1	0	Ignore	Write
0	1	1	Ignore	Ignore
1	0	0	Read	Ignore
1	0	1	Read	Inhibit
1	1	0	Write	Inhibit
1	1	1	Ignore	Ignore

Programming Word

The 8207 requires more information to operate in a wide variety of systems. The 8207 alters its timings and pin functions to operate with the 8206 ECC chip. The programming options allow the designer to use asynchronous or synchronous buses, various clock rates, various speeds and types of RAM, and others. This is detailed in Table 2.

This data is supplied to the 8207 over the PDI input pin. There are two methods of supplying this data. One is to strap the PDI pin high or low with the subsequent restrictions on your system. Table 3 shows the required system configuration. Note that your only option when strapping this pin high or low is error correction or not.

If any other configurations are required, then the programming data will have to be supplied by one or two 74LS165 type shift registers. Note that the sense of the bits in the program word change between ECC and non-ECC configurations.

Table 2a.
Non-ECC Mode Program Data Word

PD15							PD8	PD7							PD0
0	0	TM1	$\overline{PPR}$	$\overline{FFS}$	EXT	$\overline{PLS}$	CI0	CI1	$\overline{RB1}$	$\overline{RB0}$	$\overline{RFS}$	$\overline{CFS}$	SB	$\overline{SA}$	0

Program Data Bit	Name	Polarity/Function	
PD0	ECC	ECC = 0	For non-ECC mode
PD1	$\overline{SA}$	$\overline{SA}$ = 0 $\overline{SA}$ = 1	Port A is synchronous Port A is asynchronous
PD2	SB	SB = 0 SB = 1	Port B is asynchronous Port B is synchronous
PD3	$\overline{CFS}$	$\overline{CFS}$ = 0 $\overline{CFS}$ = 1	Fast-cycle iAPX 286 mode Slow-cycle iAPX 86 mode
PD4	$\overline{RFS}$	$\overline{RFS}$ = 0 $\overline{RFS}$ = 1	Fast RAM Slow RAM

Table 2a.
Non-ECC Mode Program Data Word (Continued)

PD15							PD8 PD7								PD0
0	0	TM1	$\overline{PPR}$	$\overline{FFS}$	EXT	$\overline{PLS}$	CI0	CI1	$\overline{RB1}$	$\overline{RB0}$	$\overline{RFS}$	$\overline{CFS}$	SB	$\overline{SA}$	0

Program Data Bit	Name	Polarity/Function	
PD5 PD6	$\overline{RB0}$ $\overline{RB1}$	RAM bank occupancy See Table 4	
PD7 PD8	CI1 CI0	Count interval bit 1: see Table 6 in 8207 data sheet Count interval bit 0: see Table 6 in 8207 data sheet	
PD9	$\overline{PLS}$	$\overline{PLS}$ = 0 $\overline{PLS}$ = 1	Long refresh period Short refresh period
PD10	EXT	EXT = 0 EXT = 1	Not extended Extended
PD11	$\overline{FFS}$	$\overline{FFS}$ = 0 $\overline{FFS}$ = 1	Fast CPU frequency Slow CPU frequency
PD12	$\overline{PPR}$	$\overline{PPR}$ = 0 $\overline{PPR}$ = 1	Most recently used port priority Port A preferred priority
PD13	TM1	TM1 = 0 TM1 = 1	Test mode 1 off Test mode 1 enabled
PD14	0	Reserved must be zero	
PD15	0	Reserved must be zero	

Table 2b.
ECC Mode Program Data Word

PD15							PD8 PD7								PD0
$\overline{TM2}$	RB1	RB0	PPR	FFS	$\overline{EXT}$	PLS	$\overline{CI0}$	$\overline{CI1}$	XB	$\overline{XA}$	RFS	CFS	$\overline{SB}$	SA	1

Program Data Bit	Name	Polarity/Function	
PD0	ECC	ECC = 1	ECC mode
PD1	SA	SA = 0 SA = 1	Port A is asynchronous (late AACK) Port A is synchronous (early AACK)
PD2	$\overline{SB}$	$\overline{SB}$ = 0 $\overline{SB}$ = 1	Port B is synchronous (early AACK) Port B is asynchronous (late AACK)
PD3	CFS	CFS = 0 CFS = 1	Slow-cycle iAPX 86 mode Fast-cycle iAPX 286 mode
PD4	RFS	RFS = 0 RFS = 1	Slow RAM Fast RAM
PD5	$\overline{XA}$	$\overline{XA}$ = 0 $\overline{XA}$ = 1	Multibus-compatible XACKA AACKA not multibus-compatible
PD6	XB	XB = 0 XB = 1	AACKB not multibus-compatible Multibus-compatible XACKB
PD7 PD8	CI1 CI0	Count interval bit 1: see Table 6 in 8207 data sheet Count interval bit 0: see Table 6 in 8207 data sheet	

Table 2b.
ECC Mode Program Data Word (Continued)

PD15							PD8 PD7							PD0
$\overline{TM2}$	RB1	RB0	PPR	FFS	$\overline{EXT}$	PLS	$\overline{CI0}$	$\overline{CI1}$	XB	$\overline{XA}$	RFS	CFS	$\overline{SB}$	SA	1

Program Data Bit	Name	Polarity/Function	
PD9	PLS	PLS = 0 PLS = 1	Short refresh period Long refresh period
PD10	$\overline{EXT}$	$\overline{EXT}$ = 0 $\overline{EXT}$ = 1	Master and slave EDCU Master EDCU only
PD11	FFS	FFS = 0 FFS = 1	Slow CPU frequency Fast CPU frequency
PD12	PPR	PPR = 0 PPR = 1	Port A preferred priority Most recently used port priority
PD13 PD14	RB0 RB1	RAM bank occupancy See Table 4	
PD15	TM2	TM2 = 0 TM2 = 1	Test mode 2 enabled Test mode 2 off

Table 3. 8207 Default Programming

Port A is Synchronous—has early AACK
Port B is Asynchronous—has late AACK
Fast RAM
Refresh Interval uses 236 clocks
128 Row refresh in 2 ms; 256 Row refresh in 4 ms
Fast Processor Clock Frequency (16 MHz)
"Most Recently Used" Priority Scheme
4 RAM banks occupied

Reset

If Port A is changed to an asynchronous interface (via the SA bit), then one of two precautions must be taken. Either a differentiated reset must be provided, or else software must not access the 8207 controller RAM for a short period. The 8207 is either adding or deleting internal synchronizing circuits. If a command is received during this changing, the 8207 may not perform properly. This is required only if Port A is changed to asynchronous, or if Port B is changed to synchronous.

Several of the bits in the program word determine a particular configuration of the 8207 (reference Tables 10, 11 and the 8207 Data Sheet). The bits are: CFS, CLOCK fast or slow; RFS, RAM access time fast or slow (fast refers to 100 ns—slow is everything greater); and EXT, for memory data word widths greater than 16 (22) bits. Generally speaking, CO is the fastest configuration at clock frequencies up to 16 MHz, both in the ECC or non-ECC charts. 'C3' is the fastest for 8 MHz clocks in non-ECC mode, and 'C4' is the fastest configuration when using ECC.

Take, for example, a 16 MHz 8207 clock with no error correction, a 16-bit word, and 150 ns (slow) dynamic RAMs. Table 10, in the 8207 data sheet, is used to arrive at the configuration "C1." The Timing chart Table 12 in the 8207 Data Sheet is then used to determine which clock edge to reference all timings from. The Waveforms diagrams then are used to determine the delay from the clock edge.

CHAPTER 3

RAM INTERFACE

The 8207 takes the memory addresses from the microprocessor bus and multiplexes them into row and column addresses as required by dynamic RAMs. The only hardware requirement when interfacing the 8207 to dynamic RAM are series resistors on all the RAM outputs of the 8207, and proper layout of the traces (see Intel's RAM Data Sheets or the Memory Design Handbook). This section mainly details the effects and requirements of input signals to the 8207 on the RAM array.

The 8207 contains an internal address counter used for refresh and error scrubbing (when using the 8206 EDCU) cycles. The 8207 has 18 address inputs (AIL0–AIL8 and AIH0–AIH8) which are multiplexed to form 9 address outputs (A00–A08). There are also 2 bank select (BS0, BS1) inputs for up to 4 banks of RAM. The Bank Select inputs are decoded internally to generate $\overline{RAS}$ and $\overline{CAS}$ outputs.

Refresh Interval

The 8207 supports four different refresh techniques as described in the *Refresh Options* section. In addition, the rate at which refresh cycles are performed is programmable. This is necessary because the refresh period is generated from the CLK input, which may vary over a wide range of frequencies. Programming the Cycle Fast/Slow (CFS) and Frequency Fast/Slow (FFS) bits automatically reprograms the refresh timer to generate the correct refresh interval for a clock frequency of 16, 10, 8, or 5 MHz (CFS, FFS = 11, 10, 01, and 00, respectively). For clock frequencies between those, Count Interval (CI1, CI0) programming bits allow "fine tuning" of the refresh interval. Refresh will always be done often enough to satisfy the RAM's requirements without doing refresh more often than needed and wasting memory bandwidth for all clock frequencies.

Refresh Counter

The internal refresh address counter of the 8207 contains 20 bits as organized in Figure 1.

19 18	17 16 15 14 13 12 11 10 9	8 7 6 5 4 3 2 1 0
Bank	Col addr	Row addr

Figure 1. 8207 Refresh Address Counter

In non-ECC mode, the refresh address counter does not count beyond bit 8. For standard RAMs, this will refresh 128 rows every 2 ms or 256 rows every 4 ms.

In ECC mode, the 8207 automatically checks the RAM for errors during refresh. This requires it to access each of the possible 2^{20} words of memory. The 8207 does not delete any of these bits when used with 16K and 64K dynamic RAMs. Each column would be scrubbed 4 times with 16K RAMs, and twice with 64K RAMs. This will have no detrimental effect on reliability. Banks of RAM that are not occupied, as indicated to the 8207 by the RB0, RB1 programming bits, will not be scrubbed.

Bank Selects BS0, BS1; RB0, RB1

The 8207 is designed to drive up to 88 RAMs in various configurations. The 8207 takes 2 inputs, BS0, BS1, and decodes them based on 2 programming bits, RB0, RB1, to generate the required $\overline{RAS}$/$\overline{CAS}$ strobes. Additionally, the 8207 will always recognize (not programmable) whether an access is made to the same RAM bank or to a different bank. The 8207 will interleave the accesses resulting in improved performance.

RAS and CAS Reallocation

The 8207's address lines are designed to drive up to 88 RAMs directly (through impedance matching resistors). The 4 $\overline{RAS}$ and $\overline{CAS}$ outputs drive up to 22 RAMs per bank (16 data plus 6 check bits with the 8206). Under these conditions, the 8207 will meet all RAM timing requirements. See Figure 2 for an example.

The 8207 can accommodate other configurations like a 32-bit error corrected memory system. Each bank would have 39 RAMs (32 + 7 check bits) with the total number of RAMs equal to 78. This is within the address driver's capability, but the 39 RAMs per bank exceeds the $\overline{RAS}$ and $\overline{CAS}$ driver's limits. The loading of the $\overline{RAS}$/$\overline{CAS}$ drivers should not exceed 22 RAMs per bank, otherwise critical row, column address setup, and hold times would be violated.

In order to prevent these critical timings being violated, the 8207 will re-allocate the $\overline{RAS}$ and $\overline{CAS}$ drivers based on the RB0, RB1 programming bits (see Table 4). If the RB0, RB1 bits are programmed for 2 banks, the 8207 will operate $\overline{RAS0}$ and $\overline{RAS1}$ as a pair along with $\overline{RAS2}$ and $\overline{RAS3}$, $\overline{CAS0}$ and $\overline{CAS1}$, and $\overline{CAS2}$

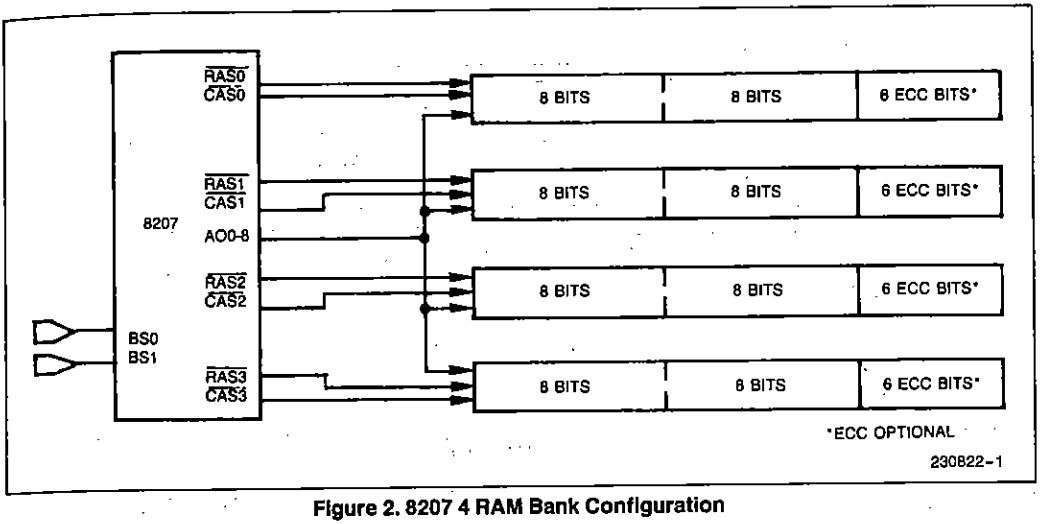

Figure 2. 8207 4 RAM Bank Configuration

and $\overline{CAS3}$. Now the address drivers would be loaded by 78 RAMs and the $\overline{RAS}/\overline{CAS}$ drivers by 20 RAMs. This relative loading is almost identical to the first case of four banks of 22 RAMs each. Drive reallocation allows a wide range of memory configurations to be used and still maintain optimal memory timings. Figure 3 shows a 32-bit non-error corrected configuration.

These programming bits do not help to qualify RAM cycles. Their purpose is to reallocate $\overline{RAS}/\overline{CAS}$ drivers. For example, if there is one bank of RAM and the bank select inputs (BS0, BS1) select any other bank and no provision is made to deselect the 8207 (via PE), the 8207 will do a RAM cycle and issue an acknowledge. This happens irregardless of the RB0, RB1 programmed value. See the *Optional RAM Bank's* section to provide for this.

Table 4. RAM Bank Selection Decoding and Word Expansion

Program Bits		Bank Input		$\overline{RAS}/\overline{CAS}$ Pair Allocation
RB1	RB0	B1	B0	
0	0	0	0	$\overline{RAS}_{0-3}$, $\overline{CAS}_{0-3}$ to Bank 0
0	0	0	1	Illegal Bank Input
0	0	1	0	Illegal Bank Input
0	0	1	1	Illegal Bank Input
0	1	0	0	$\overline{RAS}_{0,1}$, $\overline{CAS}_{0,1}$ to Bank 0
0	1	0	1	$\overline{RAS}_{2,3}$, $\overline{CAS}_{2,3}$ to Bank 1
0	1	1	0	Illegal Bank Input
0	1	1	1	Illegal Bank Input
1	0	0	0	$\overline{RAS}_0$, $\overline{CAS}_0$ to Bank 0
1	0	0	1	$\overline{RAS}_1$, $\overline{CAS}_1$ to Bank 1
1	0	1	0	$\overline{RAS}_2$, $\overline{CAS}_2$ to Bank 2
1	0	1	1	Illegal Bank Input
1	1	0	0	$\overline{RAS}_0$, $\overline{CAS}_0$ to Bank 0
1	1	0	1	$\overline{RAS}_1$, $\overline{CAS}_1$ to Bank 1
1	1	1	0	$\overline{RAS}_2$, $\overline{CAS}_2$ to Bank 2
1	1	1	1	$\overline{RAS}_3$, $\overline{CAS}_3$ to Bank 3

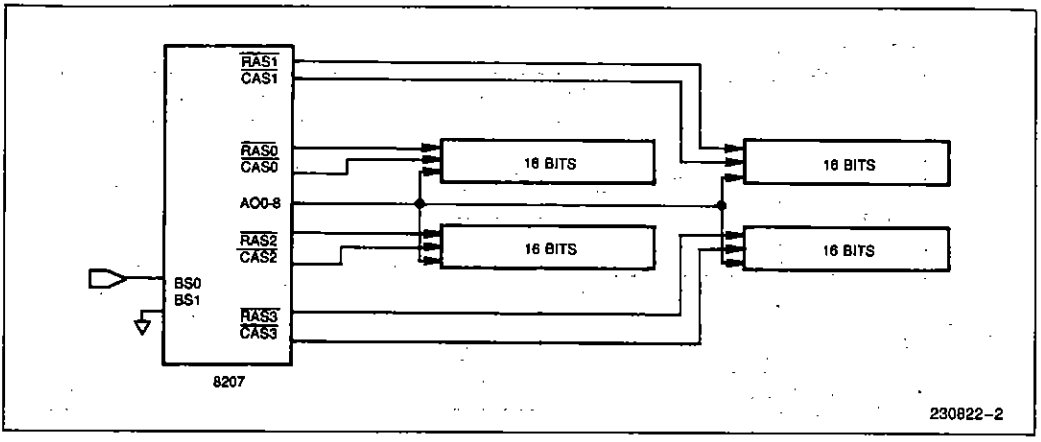

Figure 3. 8207 2 RAM Bank Configuration

Scrubbing

An additional function of the RB0, RB1 bits, besides $\overline{RAS}/\overline{CAS}$ allocation, is to inform the 8207 of how many banks are physically present. The 8207 will, during the refresh cycle, read data from a location and check to see that data and check bits are correct. If there is an error, the 8207 lengthens the refresh cycle and writes the corrected data back into RAM. Scrubbing the entire memory greatly reduces the chance of an uncorrectable error occurring. See the *Refresh* section for more detail on scrubbing.

Refresh Cycles

The 8207 performs $\overline{RAS}$ only refresh cycles in non-ECC systems. It outputs all 8207 control signals except for $\overline{CAS}$ and acknowledges. The real delay in a system due to refresh would be a fraction of that value(1). The length of the refresh cycle is always 2tRP + tRAS, and varies based upon the programmed 8207 configuration.

In error-corrected systems, the refresh cycle is actually a read cycle. The 8207 outputs a row address, then all $\overline{RAS}$ outputs go active. Next, a column address is output and then $\overline{CAS}$. The $\overline{CAS}$ output is based upon the RB0, RB1 allocation bits. Figure 4a shows the general timing for a four bank system, and Figure 4b shows a two bank system.

NOTE:

1. Measurements have shown a delay of 2–4% on program execution time compared to programs running without refresh.

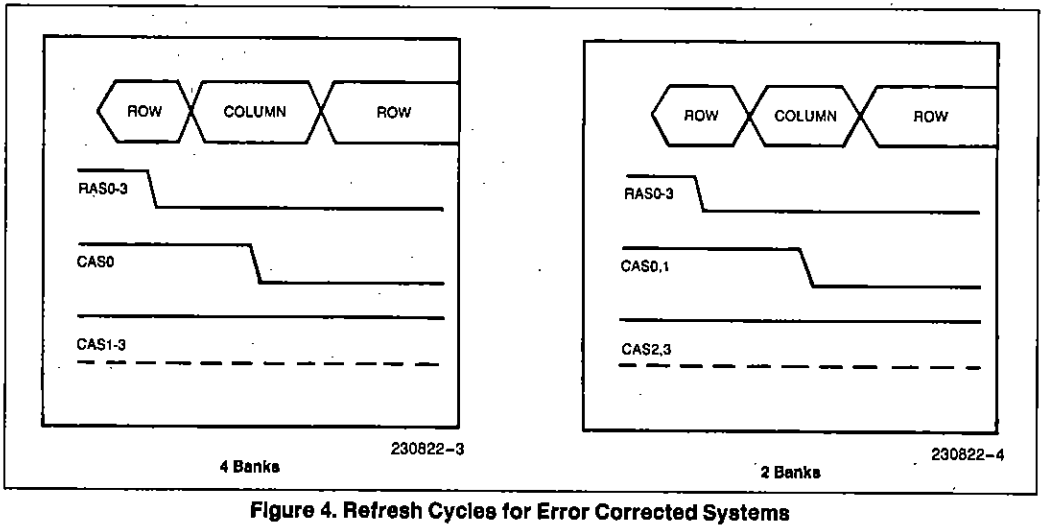

Figure 4. Refresh Cycles for Error Corrected Systems

The 8207 sends the read out word through the 8206 EDCU to check for any errors. If no errors, the refresh cycle ends. If an error is discovered, the 8207 lengthens the cycle. An error is determined if the ERROR output of the 8206 is seen active at the same edge that the 8207 issues the R/W output. The cycle is then lengthened to a RMW cycle. If the error was correctable, the corrected data is written back to the location it was read from. But, if the data is uncorrectable, the cycle is still lengthened to a RMW, but no write pulse is issued. To aid in stabilizing the RAM output data and the Error flag, pullup resistors of 10 KΩ on the data out lines are recommended.

Scrubbing removes soft errors that may accumulate until a double-bit error occurs, which would halt the system. Hard single-bit failures will not stop the system, but could slow it down. This is because read and refresh cycles lengthen to correct the data.

For large RAM arrays some form of error logging or diagnostics should be considered.

Interleaving

The term "interleaving" is often used to refer to overlapping the cycle times of multiple banks (or boards or systems) of RAMs. This has the advantage of using relatively slow cycle time banks to achieve a faster perceived cycle time at the processing unit. The drawbacks of interleaving are more logic to handle the necessary control and, for maximum performance, the program should execute sequentially through the addresses.

Dynamic RAM cycles consist of 2 parts—the RAS active time (tRAS in Dynamic RAM Data Sheets) and precharge time (tRP). The sum of these two times is roughly equal to the cycle time of the RAM. The 8207 determines how long these two periods are, based on the configuration the user picked (via the programming bits). Bank interleaving, as used by the 8207, is slightly different than the previous definition. The 8207 will overlap the precharge time of one bank with the access time of another bank. In either case, the advantage is the effective cycle time is reduced without having to use faster RAMs.

For interleaving to take place there must be more than 1 bank of RAM connected to the 8207. Interleaving is not practical with 3 banks of RAM because 3 is not a power of 2 (the 2 bank inputs BS0, BS1). So, interleaving works only for 2 or 4 banks of RAM. Note that it is easy enough to use three banks of RAM where the bank select inputs are connected to the highest-order address line. For instance, if three banks of 64K DRAMs are used in an 8086 system, and located at address OH, bank selects BS0 and BS1 would be connected to microprocessor addresses A17 and A18, respectively. Banks 0–2 would be accessed in the address ranges OH–FFFFH, 10000H–1FFFFH, and 20000H–2FFFFH, respectively. In this case, consecutive addresses are almost always in the same bank and very little interleaving can take place.

Figure 5 shows the effects on the performance of the processor with and without interleaving. In both examples, consecutive accesses to the same bank will add 1 wait state to the second access, but no wait states to consecutive accesses to different banks. Irregardless of the 8207 configuration, there will always be a minimum 1 wait state added without interleaving. Therefore, interleaving is very highly recommended!

Interleaving is accomplished by connecting the 8207's BS0, BS1 inputs to the microprocessor's low order word address lines. Each consecutive address is then located in a different bank of RAM. About 90% of memory accesses are sequential, so interleaving will occur about 90% of the time in a single port system.

In a dual port system, the advantages of interleaving are a function of the number of banks of memory. Since the memory accesses of the two ports are presumably independent, and both ports are continuously accessing memory, the 8207 arbiter will tend to interleave accesses from each port (i.e., Port A, Port B, Port A, Port B, . . .). If there are two banks of RAM interleaving will occur 50% of the time and, if there are four banks of RAM, interleaving will take place 75% of the time(1). To the extent that a single port generates a majority of memory cycles, interleaving efficiency will approach 90% as described in the previous paragraph.

NOTE:
1. Don't get confused here. The paragraph is talking about interleaving memory requests from both ports, and their probability of accessing one of the other banks of RAM where tRP has been satisfied. The 8207 will leave the RAM precharge time out if consecutive accesses go to different banks. The 8207 RAM timing logic does not care which port requests a RAM cycle.

Optional RAM Banks

Many users allow various RAM array sizes for customer options and future growth. Some care must be taken during the design to allow for this. Three items should be considered to permit optional RAM banks.

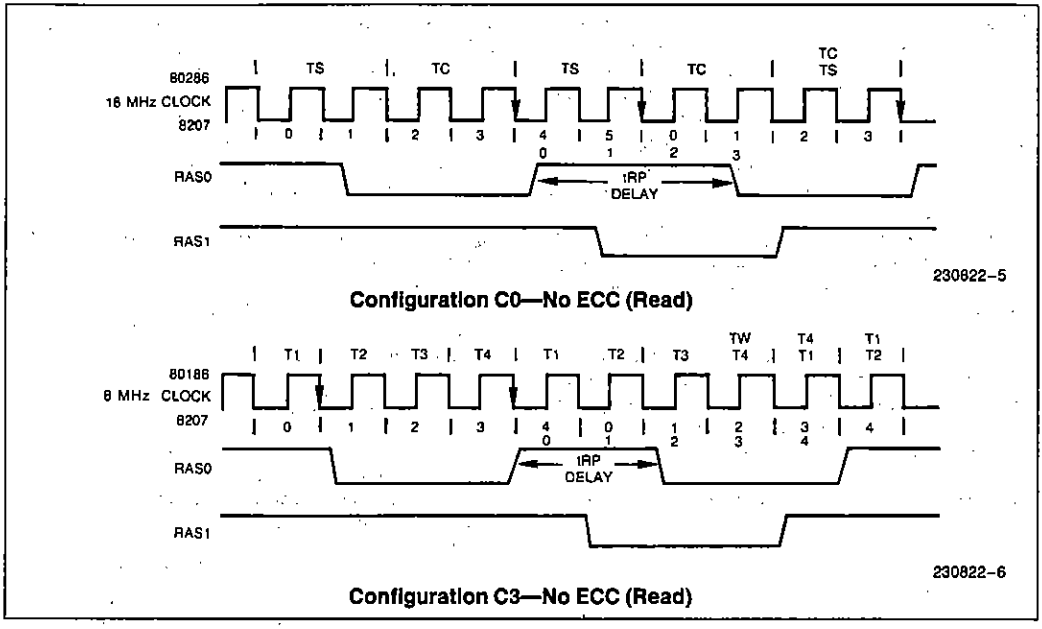

Figure 5. Processor Performance with and without Interleaving

The first item is the total RAM size. The 8207 starts a memory cycle based only upon a valid status or command and $\overline{PE}$ active. So some logic will be required to deselect the 8207 (via $\overline{PE}$) when the addressed location does not exist within the current memory size. A 7485 type magnitude comparator works well.

The second item to consider is the BS0, BS1 inputs. With one bank of RAM these inputs are tied to ground. Four banks of RAM require two address inputs. So, if the design ever needs four banks of RAM, then the BS0, BS1 inputs must be connected to address lines. Selecting a non-existant RAM bank is illegal. Figure 6 shows a non-interleaved method.

With designs using interleaving, the least significant word address lines are connected to the BS0, BS1 inputs. With two banks of RAM, A1 from the Intel processor is connected to BS0. A2 is connected to BS1, but not allowed to function until four banks are present. However, A2 must still be used since addresses increase sequentially. Two possible ways of implementing this are shown in Figure 7.

The final consideration is for the $\overline{RAS}/\overline{CAS}$ outputs. Remember that when the RB0, RB1 bits are programmed for two banks, the $\overline{RAS}$0, 1 operates in tandem (non-ECC mode/ECC mode—the $\overline{CAS}$ outputs also work in tandem). Figure 8 shows the proper layout.

Write Enables—Byte Marks

The write enable supplied by the 8207 cannot drive the RAM array directly. It is intended to be NAND with the processor supplied byte marks in a non-ECC system. In error-corrected systems, the write enable output should be inverted before being used by RAMs. Only full word read/writes are allowed in ECC systems. The changing of byte data occurs in the 8206 EDCU.

For single and dual port systems, the byte mark data (A0, $\overline{BHE}$) must be latched. The 8207 can (and will) change the input addresses midway through a RAM cycle.

Memory Warm-Up and Initialization

After programming, the 8207 performs 8 RAM warm-up cycles. The warm-up cycles are to prepare the RAMs for proper operation. If the 8207 is configured for ECC, it will then prewrite zeros into the entire array.

All $\overline{RAS}$ outputs are driven active for these cycles, once every 32 clock periods. The prewrite cycles are equivalent to write cycles, except all $\overline{RAS}$ and $\overline{CAS}$ will go active, data is generated by the 8206, and the address is generated by the 8207.

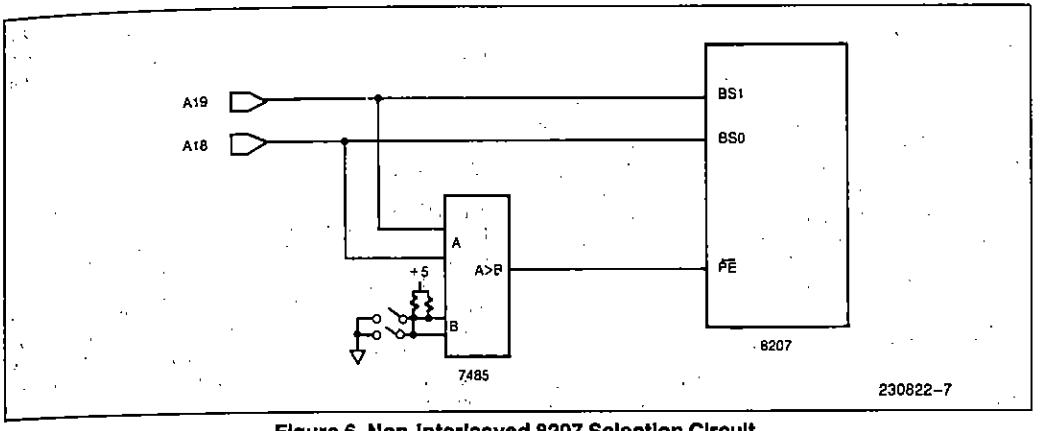

Figure 6. Non-Interleaved 8207 Selection Circuit

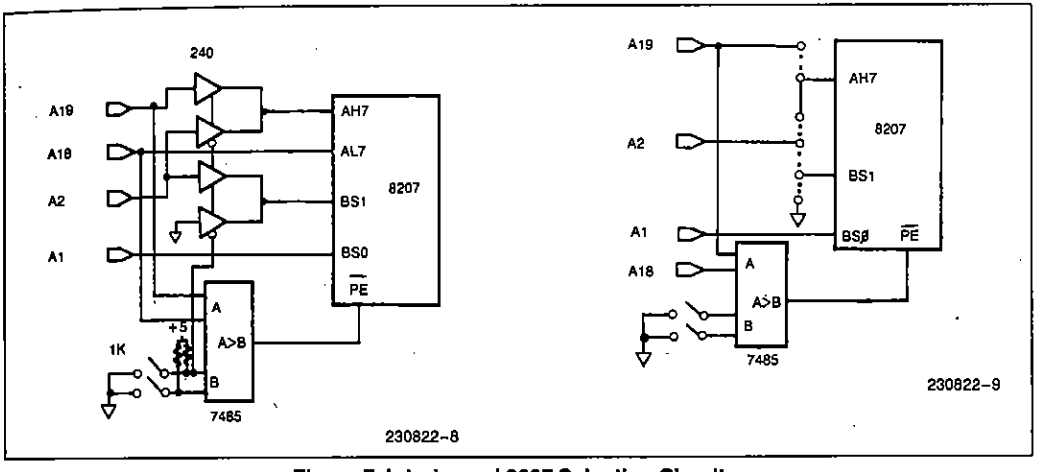

Figure 7. Interleaved 8207 Selection Circuits

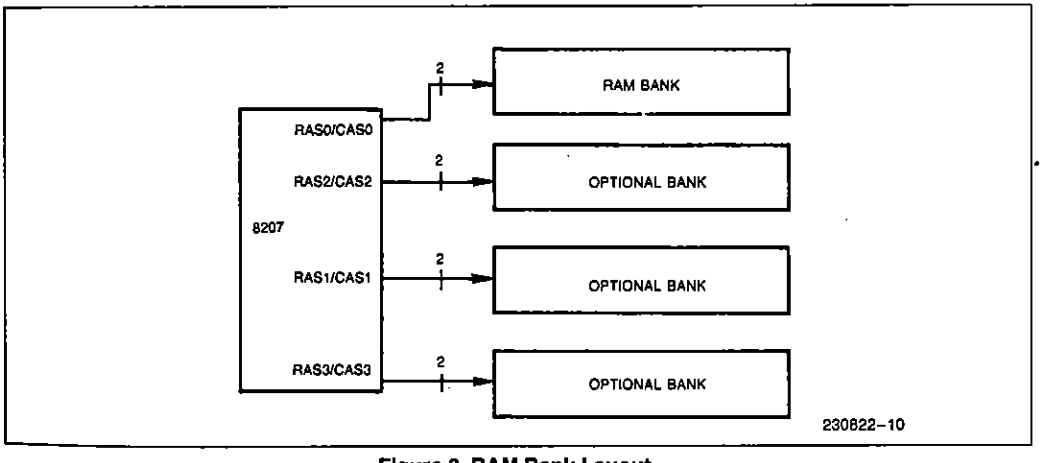

Figure 8. RAM Bank Layout

RAM Cycles/Timings

Tables 12 and 13 of the 8207 Data Sheet show on what clock edge each of the 8207 outputs are generated. This, together with the timing waveforms and A.C. parameters, allows the user to calculate the timings of the 8207 for each of its configurations. To make the job easier, Tables 14–18 of the 8207 Data Sheet precalculate dynamic RAM timings for each 8207 configuration and type of cycle. All that is required is to plug in numerical values for the 8207 parameters.

Write Cycles

The 8207 always issues WE after $\overline{CAS}$ has gone valid. These types of cycles are known as "late writes." The 8207 does this primarily to interface to the iAPX286 processor bus timings. Late writes require separate data in and data out traces to the RAM array, plus the additional drivers.

Data Latches

The 8207 is designed to meet data setup and hold times for the iAPX86 family processors when using a synchronous status interface (see Microprocessor Interface section). Other types of interfaces will require external data latches. This is because the $\overline{CAS}$ pulse is a fixed length—the user has no control (besides programming options) over lengthening $\overline{CAS}$. When $\overline{CAS}$ goes inactive, data out of the RAMs will disappear. Asynchronous interfaces should use $\overline{XACK}$ or $\overline{LAACK}$ to latch the data.

CHAPTER 4

MICROPROCESSOR INTERFACES

The 8207 is designed to be directly compatible with all Intel iAPX86, 186, 188, and 286 processors. For maximum performance, the 8207 will directly decode the status lines and operate off of the processor's clock. Additionally, the 8207 interfaces easily to other bus types that support demultiplexed address and data with separate read and write strobes.

Bus Interfaces

The 8207 easily supports either an asynchronous or synchronous command timing. The command timing can also be adjusted for various processors via the PCTL pin.

MEMORY COMMANDS

There are four inputs for each port of the 8207 that initiate a memory cycle. The input pins are $\overline{WR}$, $\overline{RD}$, PCTL, and $\overline{PE}$. The first three inputs connect directly to the iAPX 86, 88, 186, 188 $\overline{S0}$–$\overline{S2}$ outputs, respectively. For the 80286, the same connections are used except that PCTL is tied to ground. In all configurations $\overline{PE}$ is decoded from the address bus. Multibus type commands use the same input setup as the 80286.

COMMAND/STATUS INTERFACE

The status interface for the 80186 and the 80286 differ both in timing and meaning. The 8207 can be optimized for either processor by programming the PCTL input pin at RESET time. $\overline{S2}$ in 80186 systems, connects directly to PCTL. When the processor is reset it drives $\overline{S2}$ high for one clock, then tristates it. A pullup resistor to +5 will program the PCTL input for the 80186 status interface when RESET goes inactive. A pullup is required only if no component has this pullup internally.

To optimize the 8207 for the 80286 interface, PCTL is tied to ground and not used in 80286 systems. Multibus commands are similar in meaning to the 80286 status interface, and are programmed the same way. In Multibus type systems, PCTL can be used as an inhibit to allow shadow memory. PCTL would be driven high, when required, to prevent the 8207 from performing a memory cycle. It would be connected to the Multibus INH pin through an inverter.

SYNCHRONOUS/ASYNCHRONOUS COMMANDS

Each port of the 8207 can be configured to accept either a synchronous or asynchronous (via programming bits) memory request. Minimum memory request decode time (and maximum performance) is achieved using a synchronous status interface. This type of interface to the processor requires no logic for the user to implement.

An asynchronous interface is used with Multibus bus interfaces when the setup and hold times of the memory commands cannot be guaranteed. Synchronizers are added to the inputs and will require up to two clocks for the 8207 to recognize the command. It should be obvious that better performance will result if the 8207's clock is run as fast as possible.

Figure 2 of the 8207 Data Sheet shows various combinations of interfaces. The additional logic for the asynchronous interfaces is used to either lengthen the command width, to meet the minimum 8207 spec, or to make sure the command does not arrive too soon before the address has stabilized.

PORT ENABLE

The $\overline{PE}$ inputs serve to qualify a memory request. A RAM cycle, once started, cannot be stopped. A RAM cycle starts if $\overline{PE}$ is seen active at the proper clock edge and a valid command is recognized. If $\overline{PE}$ is activated after a command has gone active and inactive, no cycle will start.

Types of logic that work well are 74138 and 7485. $\overline{PE}$ should be valid as much as possible before the command arrives because, as the address bus switches and settles, glitches on $\overline{PE}$ could either: disqualify a memory cycle; delay a memory cycle; or start a memory cycle when none should have. Refer to the Port Interface Waveforms in the Data Sheet. If Port Enable is not seen active by the next or same clock edge, no memory cycle will occur unless the command is removed and brought active again.

Back to Back Commands

Holding the $\overline{RD}$, $\overline{WR}$ inputs active will not generate continuous memory cycles. Memory commands must go inactive for at least one clock period before another memory request at that port will be considered valid. Holding the inputs active will not keep the other port from gaining access to the RAM. The only signal that can prevent the other port's gaining access to the RAM is LOCK.

Address Inputs (and LOCK)

Two pins control the address inputs on the 8207, MUX and LEN. Neither are used for single port 8086 based systems. MUX is used for dual port configurations, and LEN is used for single and dual port 80286 based systems. MUX is used to gate the proper ports addresses to the 8207. If the output is high, Port A is selected. If it is low, Port B is selected.

The cross coupled NAND gates, shown in the 8207 Data Sheet (Figure 3), are used to minimize contention when switching address buses. Use of a single inverter would have both outputs enabled simultaneously for a short period. The cross coupled hand gates allow only one output enabled.

MUX also allows the single LOCK input to be multiplexed between ports. Figure 9 shows how to multiplex the LOCK input for dual port systems. See the LOCK section for more information.

MUX TIMING

The MUX output is optimized by the Port Arbitration scheme, which is selected in the program word. Figure 10 shows the effects on memory bandwidth with the different schemes. Port A Preferred optimizes consecutive cycles for Port A. Consecutive Port B cycles have at least 1 clock added to their cycle time. There would be no MUX delays for any Port A request.

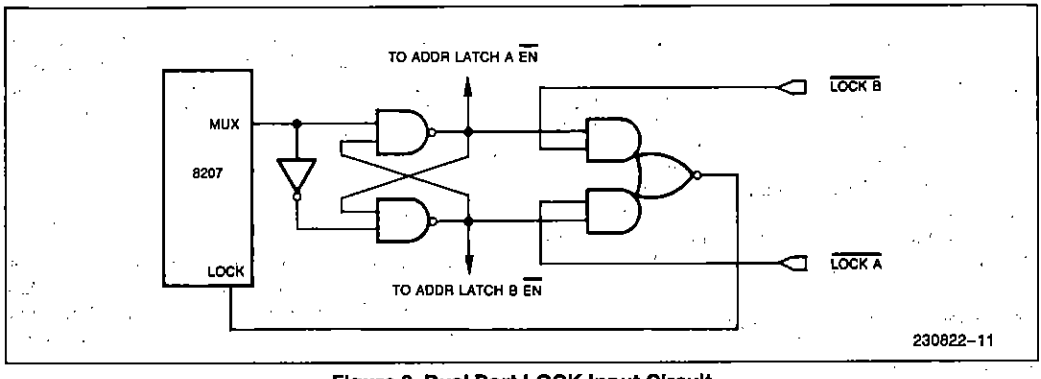

Figure 9. Dual Port LOCK Input Circuit

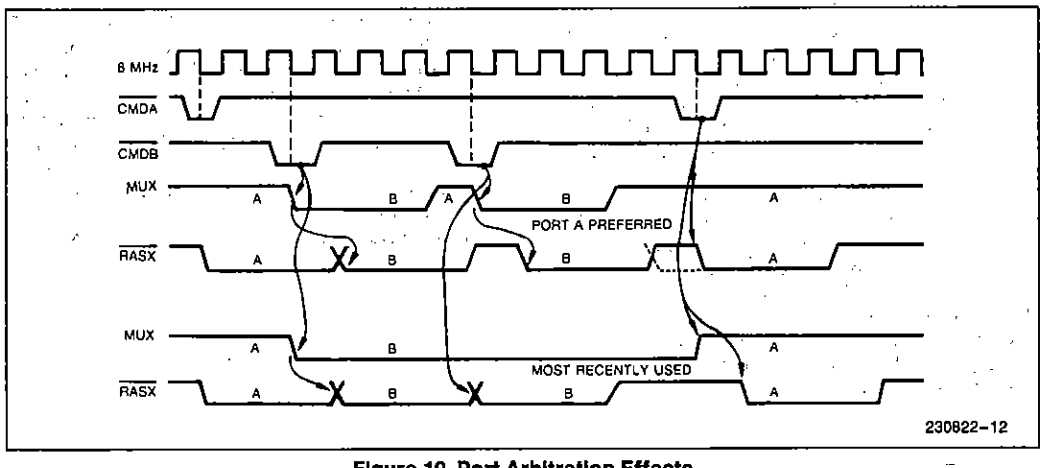

Figure 10. Port Arbitration Effects

The Most Recently Used scheme allows either port to generate consecutive cycles without any MUX delays. The first memory cycle for each port would have the 1 clock delay. But all others would not.

With either scheme, if both ports request the memory at their top speed, the 8207 will interleave the requests; Port A, Port B, Port A, Refresh, Port B.

LEN

LEN is used to hold the 80286 addresses when the 8207 cannot respond immediately. The 8207 will require a separate address latch, with the ALE input replaced with LEN. LEN optimizes the address setup and hold times for the 8207.

LEN goes from high to low when a valid 8207 command is recognized, which latches the 80286 address. This transition of LEN is independent of a memory cycle starting. The low to high transition will occur in the middle of a memory cycle so that the next address will be admitted and subsequently latched.

If Port B is to interface to an 80286 with the synchronous status interface, then LEN must be created using external logic. Figure 11 shows the equivalent 8207 circuit for Port B.

LOCK

The LOCK input allows each port uninterrupted access to memory. It does this by not permitting MUX to switch. It is not intended as a means to improve throughput of one of the ports. To do so is at the designer's risk(1). Obviously, LOCK is only used in dual port systems. The 8207 interprets LOCK as originating from the port that MUX is indicating.

NOTE:
1. The 8207 will not mulfunction if this is done. This is a system level concern. For example, a time dependent process may fail if the other port holds LOCK active, preventing its access of memory and relinquishing the bus.

LOCK from the 8086 may be connected directly to the 8207 or to the multiplexing logic. The 8207 requires additional logic when interfaced to an 80286. Figure 12 shows both the synchronous and asynchronous circuitry.

For 16 MHz operation, the 8207 ignores the LOCK input during the clock period that MUX switched. During 8 MHz operation, the 8207 will see LOCK as being active during the clock period when MUX switches.

The LOCK issued in Multibus bus systems may not be compatible with the 8207. The 8207 references LOCK from the beginning of a cycle, while Multibus references LOCK from the end of a cycle. The Multibus LOCK can be used if it meets the 8207 requirements. If the LOCK timing cannot be guaranteed, then additional logic is necessary. The logic would issue LOCK whenever a Multibus command is recognized. The drawback to this is that MUX cannot switch during the RAM cycle. This would delay the other port's memory access by one or two clocks.

DEADLOCK

The designer should ensure that a deadlock hazard has not been created in the design. The simple interfaces shown previously will not create a deadlock condition when the 8207 controls all system memory. If LOCK is issued by both ports, then the above logic would need to be modified to remove LOCK.

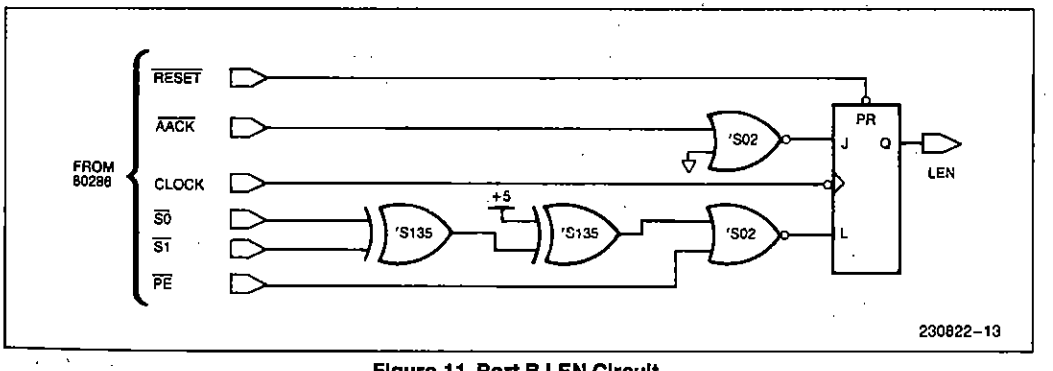

Figure 11. Port B LEN Circuit

Figure 13 shows an illustration of the problem with a single LOCK input.

Suppose the 8207 starts a locked string transfer for the processor. The Multibus bus port requests a memory cycle but must wait for the processor to remove LOCK. But the processor must access Multibus as part of the locked string transfer. We now have a deadlock. The solution is to force LOCK inactive whenever an access is made to non-8207 memory by the processor. By doing this we have now violated the purpose of LOCK, since the Multibus port could change data. Another solution is to ensure that locked data does not exist in physically separate memory.

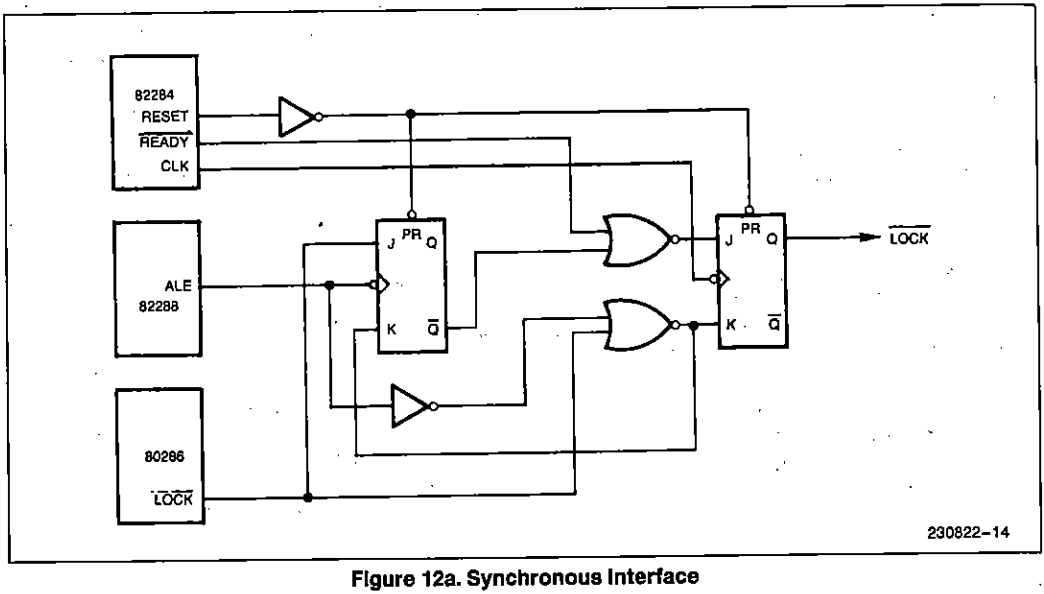

Figure 12a. Synchronous Interface

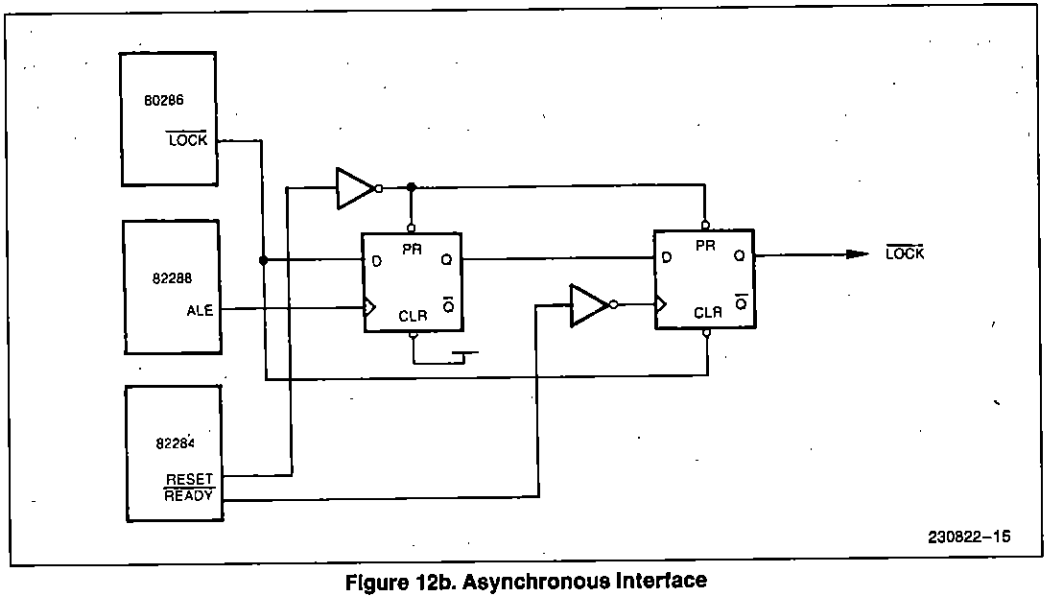

Figure 12b. Asynchronous Interface

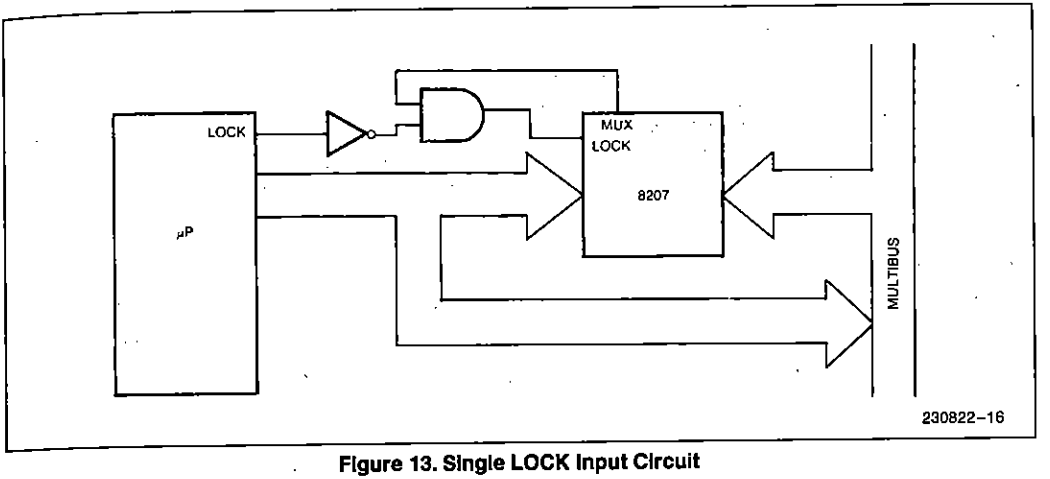

Figure 13. Single LOCK Input Circuit

8207 Acknowledges

The 8207 in non-ECC mode has two active acknowledges per port, $\overline{AACK}$ and $\overline{XACK}$. The $\overline{AACK}$ output is configured into either an "early" or "late" $\overline{AACK}$ based on the SA, SB bits in the program data word. In ECC systems there is one Acknowledge per port, and it is configured to any one of the three ($\overline{EAACK}$, $\overline{LAACK}$ or $\overline{XACK}$) by the programming bits.

The $\overline{AACK}$ pin is optimized for either the 80286 or the 8086, based upon the CFS programming bit (fast = 80286; slow = 8086). $\overline{XACK}$ conforms to the Multibus bus specification. $\overline{XACK}$ requires a tri-state buffer and must not drive the bus directly.

In synchronous systems, $\overline{XACK}$ will not go active if the memory command is removed prior to the clock period that issues $\overline{XACK}$. In asynchronous systems, the $\overline{AACK}$ pin can also serve as an advanced RAM cycle timing indicator.

Data out, in synchronous systems, should not have to be latched. The 8207 was designed to meet the data setup and hold times of Intel processors, the 8086 family, and the 80286. In asynchronous systems, the 8207 will remove data before the processor recognizes the Acknowledge ($\overline{LAACK}$ or $\overline{XACK}$). In these systems, the data should be latched with transparent type latches (Intel 8282/8283).

Output Data Control

NON-ECC

In single port systems, Intel processors supply the necessary timing signals to control the input or output of data to the RAMs. These control signals are $\overline{DEN}$ and DT/$\overline{R}$. Refer to the microprocessor handbook for their explanation. If these signals are not available, then PSEN and $\overline{DBM}$ provide the same function. They can be used directly to control the 8286/8287 bus drivers of the 8207.

Because of the single set of data in/out pins of the RAMs, data must be multiplexed between the two ports in dual port systems. The 8207 provides two outputs for contention-free switching. PSEL operates the same as the MUX output, in that a high selects Port A and a low selects Port B. PSEN acts to enable the selected port. The timing is shown in the 8207 Data Sheet, Port *Switching Timing* section.

The easiest means of using PSEL and PSEN is shown in Figure 14. At no time will both ports be enabled simultaneously.

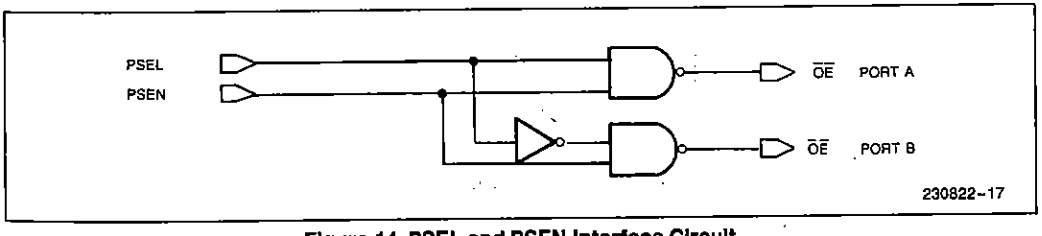

Figure 14. PSEL and PSEN Interface Circuit

Data Bus—Single Port

Recall that the 8207 always performs a late write cycle and that this requires separate data in and out buses. One option for the data bus is shown in Figure 3 of the 8207 Data Sheet. It requires separate data in and out traces on the processor board.

The second option is to keep the processor's combined data, but separate the data at the 8207 RAM. This is shown in Figure 15.

Data Bus—Dual Port

NON-ECC

The multiplexed data of the 8207 RAM must be kept isolated so that an access by one port does not affect another port. Figure 16 illustrates the control logic.

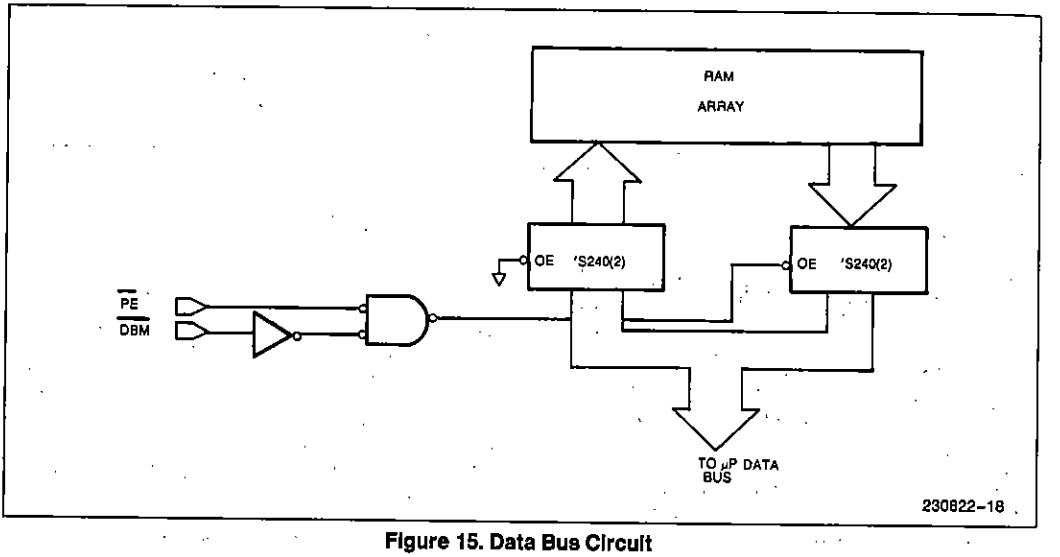

Figure 15. Data Bus Circuit

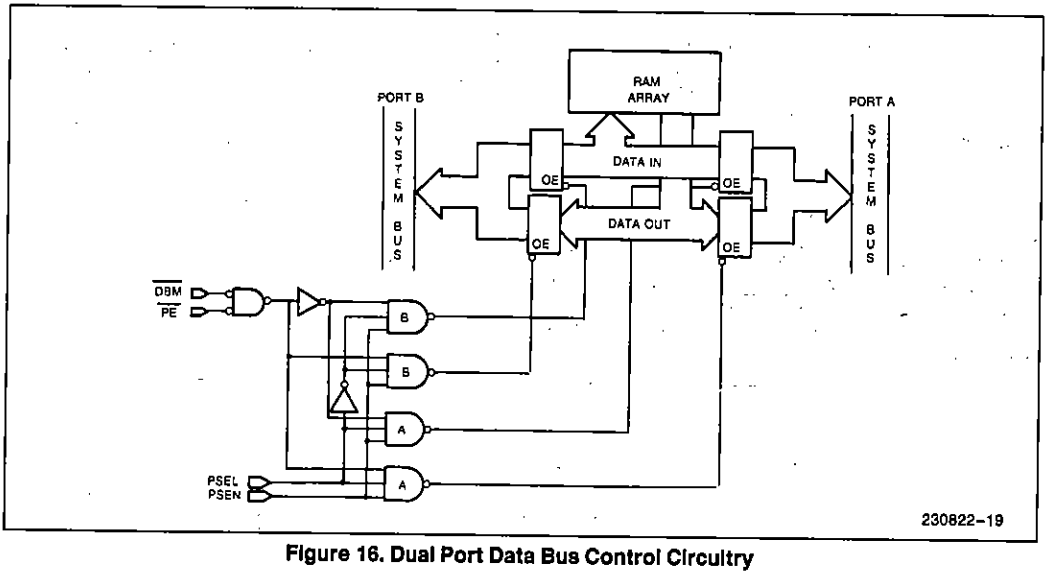

Figure 16. Dual Port Data Bus Control Circuitry

CHAPTER 5

8207 WITH ECC (8206)

This section points out the proper control of the 8206 EDCU by the 8207.

The 8207 performs error correction during read and refresh cycles (scrubbing), and initializes memory after power up to prevent false errors from causing interrupts to the processor. Since the 8207 must refresh RAM, performing scrubbing during refresh allows it to be accomplished without any additional performance penalty. Upon detection of a correctable error during scrubbing, the RAM refresh cycle is lengthened slightly to permit the 8206 to correct the error and for the corrected word to be rewritten into memory. Uncorrectable errors detected during scrubbing are ignored, since the processor may never access that memory location.

Correctable errors detected during a memory read cycle are corrected immediately and written back into memory.

Synchronous/Asynchronous Buses

The many types of configurations that are supported by the 8207/8206 combination can be broken down into two classes: ECC for synchronous or for asynchronous buses.

In synchronous bus systems, performance is optimized for processor throughput. In asynchronous buses, performance is optimized to get valid data onto the bus as quickly as possible (Multibus). While possible to optimize the 8207/8206 for processor throughput in Multibus systems, it is not Multibus compatible. The performance optimization is selected via the XA/XB and SA/SB programming bits.

When optimized for processor throughput, an advanced acknowledge ($\overline{AACK}$—early or late) is issued at some point (based on the type of processor) so that data will be valid when the processor needs it.

When optimized for quick data access, an $\overline{XACK}$ is issued as soon as valid data is known to exist. If the data was invalid (based on the ERROR flag), then the $\overline{XACK}$ is delayed until the 8206 corrects the data and the data is on the bus.

The first example is known as "correct always" mode. The 8206 $\overline{CRCT}$ pin is tied to ground and the 8206 requires time to do the correction. Figure 17 shows this implementation. The quick data access method is known as "correct on error." The CRCT pin is tied to the R/$\overline{W}$ output of the 8207. When CRCT is high, the 8206 does not do correction, but still checks the data.

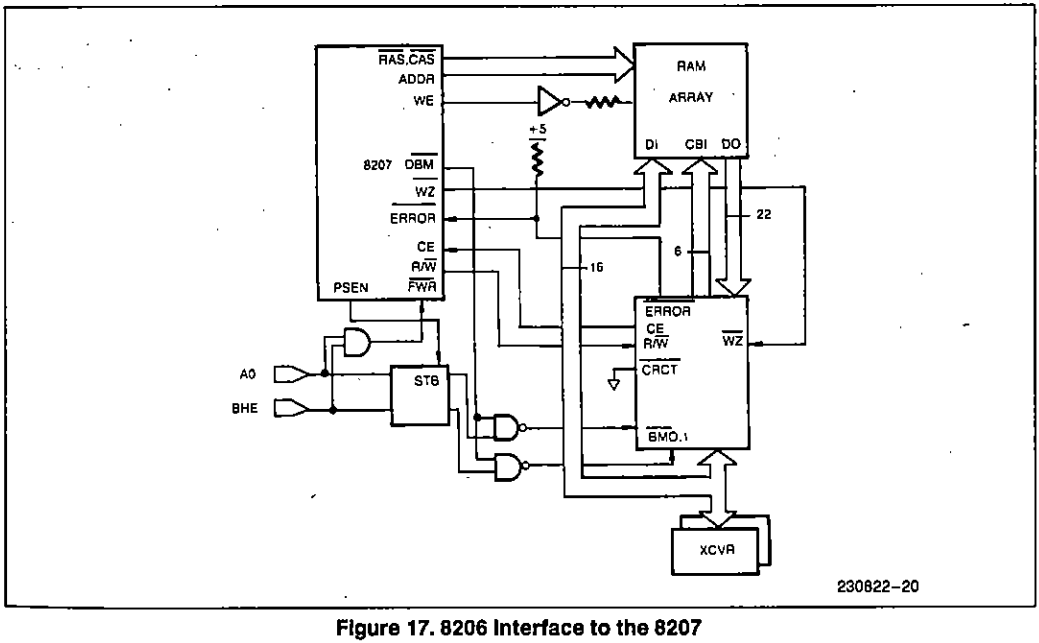

Figure 17. 8206 Interface to the 8207

This delay is typically half of the first. If an error happens, the cycle becomes a RMW and $\overline{XACK}$ is delayed slightly so that data can be corrected.

The correct on error mode is of no real benefit to non-Multibus users. The earliest acknowledge (EAACK) is delayed by one clock to allow for the delays through the 8206. This imposes a 1 wait state delay.

Byte Marks

The only real difference to the 8207 system when adding the 8206 is the treatment of byte writes. Because the encoded check bits apply only to a whole word (including check bits), byte writes must not be permitted at the RAM. Instead, the altering of byte data is done at the 8206. The byte marks previously sent to RAM are now sent to the 8206. These byte marks must also qualify the output enables of the data drivers.

The $\overline{DBM}$ output of the 8207 is meant to be nanded with the processor's byte marks. This output is activated only on reads or refreshes. On write cycles, this output stays high which would force the 8206 byte mark input low. When low, the internal 8206 data out buffers are tristated so that new data may be gated into the device.

Read Modify Writes—ECC

A RMW cycle occurs whenever a processor wants to do byte writes or when the 8207 has detected an error during read or refresh (scrubbing) cycles. A byte write is detected by the $\overline{FWR}$ input to the 8207 and is based on the processor supplied byte marks.

At the start of a RMW cycle, $\overline{DBM}$ stays high, which, when qualified with the byte marks, will enable the data out buffer of the 8206 for the unmodified byte, and tristates the buffer for the new byte; R/$\overline{W}$ is high, which tells the 8206 to do error detection and correcting (if $\overline{CRCT}$ is low). The 8206 can latch data and check bits from the RAM via the STB input, but the 8207 does not use this feature. Instead, the 8207 keeps $\overline{CAS}$ active the entire length of the RMW cycle to hold data at the 8206. The new byte data from the processor goes to the 8206 and to the RAM. The 8207 would have corrected any errors just read, so the old and new bytes of data, plus their check bits, are available at the RAM, and the 8207 generates a write pulse. The data driver for the unmodified byte must not have been enabled, otherwise erroneous data would be written to RAM and possibly made valid (if it was stable) by the 8206.

Data Buffer Control—ECC

The control of the data buffers is essentially the same as in non-ECC systems, with a few exceptions. The processor's byte marks must now qualify the output enable logic. The reason was described earlier in the RMW section. This applies to both single and dual port configurations. A refresh cycle outputs all the control signals that a read cycle will, except for an acknowledge. If complete buffer control is left to the 8207, then it would occasionally (during refreshes) put data on the processor bus. The $\overline{DEN}$ and DT/$\overline{R}$ signals must be qualified by the $\overline{PE}$ input. $\overline{PE}$ would have to be latched for the entire cycle by PSEN.

Test Modes

Neither of the two test modes of the 8207 are to be used in a design. Both test modes reset the refresh address counter to a specific value, which interrupts the refresh sequence and causes loss of data.

In error corrected systems, a reset pulse causes the 8207/8206 to write over the entire RAM array. Test Mode 2 appears to bypass the prewrite sequence. But, the refresh counter is reset to a value of 1F7 (H). So, besides interrupting the refresh sequence, the 8207 still prewrites the 8 locations specified by the counter.

To not overwrite the RAM data, the 8207 RESET will have to be isolated from the system reset logic in ECC systems.

APPENDIX A
8207 PERFORMANCE

The following performance charts were based upon Figure 3 in the 8207 Data Sheet. The charts show the performance of a single cycle with no precharge, refresh, port switching, or arbitration delays.

The read access calculations are: the margin between the 8207 starting a memory cycle to data valid at the processor − 8207 RAS or CAS from clock delay − DRAM RAS or CAS access − 8286 propagation delay − processor set-up.

Assume the RAS/CAS drivers are loaded with 150 pF, and the 8286 is driving a 300 pF data bus.

80286 (example)

RAS Access: 3TCLCL − 8207 TCLRSL − 2118 tRAC − 8286 TIVOV − 80286 t8
= (3)62.5 − 35 max − 100 max − 22 − 10
= 20 ns

80186 (example)

CAS Access: 2 TCLCL − 8207 TCLCSL − DRAM tCAC − 8286 TIVOV − 80186 TDVCL
= (2)125 − 115 max − 85 max − 22 − 20
= 8 ns

8207 Performance (EDC synchronous status interface)

Table 5a. Wait States for Different μP and RAM Combinations

Wait States at Full CPU Speed		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	1-RD, WR 3-Byte WR C0(3)	1-RD, WR 3-Byte WR C0	2-Read 1-Write 3-Byte WR C2	Not Compatible with RAM Parameters(1)
80186, 8086/88-2	8 MHz	1-RD, WR 3-Byte WR C4	1-RD, WR 3-Byte WR C4	1-RD, WR 3-Byte WR C4	
8086/88	5 MHz	1 C6	1 C6	1 C6	1-RD, WR 3-Byte WR C4

Table 5b. μP Clock Frequency for Different μP and RAM Combinations

Maximum Frequency for One Wait-State(4)		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	Full Speed		7.3 MHz C0	6 MHz C0
80286, 8086/88-2	8 MHz				7 MHz C4
8086/88	5 MHz				

8207 Performance (Non-EDC synchronous status interface)

Table 6a. Wait States for Different μP and RAM Combinations

Wait States at Full CPU Speed		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	0 C0(3)	1-Read 0-Write C1	1-Read 0-Write C1	Not Compatible with RAM Parameters(1)
80186, 8086/88-2	8 MHz	0 C3	0 C3	0(2) C3	
8086/88	5 MHz	0 C3	0 C3	0 C3	0 C3

Table 6b. μP Clock Frequency for Different μP and RAM Combinations

Maximum Frequency for No Wait-State(4)		RAM Speed			
CPU	Freq	100 ns	120 ns	150 ns	200 ns
80286	8 MHz	Full Speed	7 MHz	6 MHz	5.3 MHz
80186, 8086/88-2	8 MHz		Full Speed		7 MHz
8086/88	5 MHz				

NOTES:
1. The DRAM tRAH parameter is not satisfied.
2. 150 ns 64K DRAMs with tCAC = 100 ns won't run with 0 wait-states.
3. Numbers in lower right corners are the programmed configurations of the 8207.
4. To meet read access time.

8207 Performance (MULTIBUS Interface)

This is an *asynchronous, command interface.* Worst case data and transfer acknowledge (XACK#) delays. Including synchronization and data buffer delays, are:

Table 7a. Non-EDC System

	RAM Speed			
	100 ns	120 ns	150 ns	200 ns
Data Access Time	289 ns	299 ns	322 ns	380 ns
XACK# Access Time	333 ns			450 ns

Table 7b. EDC System

	RAM Speed			
	100 ns	**120 ns**	**150 ns**	**200 ns**
Data Access Time (Read)	359 ns (324 ns)(1)	369 ns (334 ns)	392 ns (357 ns)	450 ns (415 ns)
XACK# Access Time	400 ns-RD, WR 588 ns-Byte Write			520 ns-RD, WR 806 ns-Byte WR

NOTE:
1. Numbers in parentheses are for when 8206 is in check-only mode (8206 doesn't do error correction until after an error is detected).